AMENDMENTS TO THE SPECIFICATION:

Please amend the paragraph beginning at page 1, line 3, as follows:

Technical ~~field~~ Field

Please amend the paragraph beginning at page 1, line 10, as follows:

Background and ~~summary~~ Summary ~~of the invention~~

Please amend the paragraph beginning at page 1, line 11, as follows:

The amount of traffic on the web, i.e., in the Internet, is constantly increasing. Much of the traffic is produced by users who may not be familiar with the implementation of web software and its interfaces. Many software products have developed bad reputations for their cryptic error messages that are hard to understand and resolve. Consequently, web site user interfaces ~~are bad in handling~~ could better handle problem situations.

Please amend the paragraph beginning at page 2, line 24, as follows:

The goal of Enterprise Management is to provide a focal point for surveillance of the computing environment in a company. ~~The total success~~ An objective of an Enterprise Management Framework is to provide a solution automatically executed on all fatal events that has an impact on a company's IT based system. This scenario is far away for Enterprise Management but in the discipline of Network Management there are many solutions, some of which are patented and referred to in this application, since Network Management is a very mature discipline.

Please amend the paragraph beginning at page 7, line 20, as follows:

No event is sent to the Management System regarding this type of error. This means that the companies have to rely on reports from customers to become aware that their application has failed. The customer could easily choose another company to supply services if ~~loosing~~ the customer loses confidence in the previous company that had a. malfunctioning WEB application. It is known that for a company entering the WEB, a poorly functioning WEB site is very dangerous as the competitor is "only one click away." In this case, it could be better to shut down the WEB site and declare it to be in maintenance mode, to declare to customers that the company controls its business critical WEB applications.

Please amend the paragraph beginning at page 8, line 18, as follows:

The present invention provides a method to automatically detect objects that unexpectedly ~~disrupts~~ disrupt execution in an object oriented software application. This intrusive method contains a registration of objects, i.e., class instances such as Java classes or C++ classes, in a persistent location so that the codes for the object may be classified. A monitor supervising the persistent location is able to detect expired objects, i.e., objects that have remained on the persistent location too long without having been properly removed therefrom, and report them as an alarm to the Enterprise Management Framework that there is an operational failure for the object that has not been deregistered. The registration is connected to a deregistration of the object when the object terminates. The deregistration removes the registered object from the persistent location. The registration will also handle objects that are active for a longer time period

as the monitor is able to supervise the object for activity. To be able to track down a failure, methods for tracing object inheritance and context handling ~~are~~ is provided ~~in the method~~ by identifying the objects that have not been properly deregistered.

Please amend the paragraph beginning at page 11, line 16, as follows:

When an application 7 on a server 3 connected to the Internet 1 is started, some computer program section or method can be run on a programming language, such as Java, as is assumed herein, the program section then being a Java scriptlet. In the code of the program section, instructions are added for storing basic or initiating information relevant to the application in a persistent location 601, see Fig. 6, as will be described below. Special classes called, for example, WAM, Module_name and Class_name may be defined which is only used for supervision purposes.

Please amend the paragraph beginning at page 12, line 1, as follows:

When ~~then~~ a user from a network terminal 5 connected to the Internet 1 logs into the already started application 7, a program section comprised in the application is started and then a file is sent to the terminal and there generates, through a browser-program, an image or page shown on the display or monitor of the terminal. On the page, various fields for entering information can be provided and simple "button" fields which can be marked by the user by moving some pointing means such as a computer mouse and "clicking" on it, i.e., pressing a button on the mouse. For most of such operations, program sections or scriptlets are started which perform the specific steps of the service or application, one scriptlet being specially provided for or linked to each such operation.

Please amend the paragraph beginning at page 12, line 14, as follows:

In at least some of the scriptlets, which generally are shorter or longer sections of code, special instructions are inserted for allowing a supervision of the execution of the program section by entering information in the persistent location. Thus for each such program section an instruction is added first in the executable code for creating a special object, an instance of a special class or example called WAM, that is designed only for the supervision of the execution of the program section, see the items "Object X" and "Object Y" in the schematic view of Fig. 1. Other added instructions invoke special methods are designed for the supervision of the program sections, these methods called an API (Application Programming Interface) and illustrated by the blocks 102. The special methods can include a method Register for registering an object in the persistent location, a method Deregister for cancelling the registering an object and methods for storing information on the specific execution of program sections, such as alarm parameters and messages.

Please amend the paragraph beginning at page 13, line 7, as follows:

In an instruction added directly after the instruction for creating the special object, this object is registered by invoking the method Register and then information related to the object is stored in the persistent location. In further added instructions messages and information related thereto and alarm parameters and information related thereto can be stored in the persistent location. Alarm parameters are metrics or numeral values related to the object. Finally, in a final instruction of the code section, i.e., in an instruction

inserted as the last one before terminating the program section, the special object is deregistered by invoking the method Deregister. Then the information stored in the persistent location and related to the object is cancelled. Also, other possible information, ~~of~~ e.g., alarm parameters, used in the program section ~~is~~ may be cancelled by the same method.

Please amend the paragraph beginning at page 14, line 21, as follows:

The organization of the memory area 601 called the persistent location is illustrated in Fig. 6. The persistent location 601 comprises six subareas, a node section 603, a module section 605, a class section 607, a message section 609, a parameter section 611 and an object section 613. Each of the subareas except the node section ~~601~~ 603 is organized like a register or table in a database and comprises a plurality of slots 615, 617, 619, 621 and 623 respectively, each slot having an order number or sequential address and in each slot information of a single item being stored.

Please amend the paragraph beginning at page 17, line 16, as follows:

In the sixth subarea, the object section 613, information on the instances of a class that are currently being executed or that have recently been executed is stored. ~~In each~~ Each slot 623 of the object section ~~there are~~ may include (perhaps with other fields) a field for storing a pointer to class, a field for storing a unique register key, as will be described below, for printing, a field for an identification of a context for printing, a field for storing a time of registering, a field for storing information of the remaining time until expiration, a field for an activity counter, a field for storing the value of an expiration

time period to be used when initiating or re-initiating the information in the field for information of the remaining time and ~~finally~~ a field for storing an indication whether the object is in a running or lost state. An object is said to be running or in a running state if there has been activity recorded for the object in the time period of length equal to the expiration time up to the current time. An object is said to be lost or in a lost state if there has been no activity recorded for the object during the time period of length equal to the expiration time up to the current time.

Please amend the paragraph beginning at page 20, line 4, as follows:

If it instead is determined in the block 205 that a free slot 623 exists, a block 210 is executed, in which information on the current object is stored, see the description of the contents of a slot 623 given above. Then information can be stored in the slot. Thus in the block 210 the current time, i.e., the time of storing information in the slot, and a value of an expiration time period. In the next block 212 a unique registration key or register key is created by the method and is stored in the slot. The register key is to be used when tracing object creation. The information stored in an object section slot ~~must always include at least such as~~ <u>includes</u> some information, is e.g., the register key, identifying the object and an expiration period. Furthermore, other information, not illustrated in Fig. 2, can be stored in the selected slot 623 such information, e.g., including information identifying a link that is set-up to the class 607 of which the object is an instance. In the slot 617 of the class section a count of instances, which are currently being executed, is stored. In addition, in the slot the field including information on the time period

remaining until expiration is set to be equal to the expiration time and the field for storing the activity counter is reset, i.e., is set to be equal to zero.

Please amend the paragraph beginning at page 21, line 19, as follows:

The process when scanning the object section is illustrated by the flow diagram of Fig. 3. In a first block 300 a pointer, not shown, is set to indicated the first object section slot 623. Then it is determined in a block 302 whether the block indicated the pointer, exists. If it does not exist, it means that all slots have been handled and the procedure exits as indicated by the arrow 308. Otherwise it determined in a block 304 whether the slot indicated the pointer is an active slot, i.e., whether information for an object being currently executed is stored therein. The determination is made by testing some marking field, not shown, of the slot. If it is determined that the slot is not active, a block 306 is executed in which the pointer is set indicate the next slot of the object section 613. After block 306, the block ~~320~~ 302 is again executed, as described above.

Please amend the paragraph beginning at page 22, line 9, as follows:

If it is determined in the block 304 that the current ~~block~~ slot is active, the information stored in the slot is accessed. Then in a block 310 the information or value of the time remaining until expiration is decremented by the time period elapsed from the previous scanning of the object section 613. In the next block 312 it is determined, whether the object has expired. This can be done by testing the result of the decrementing in the previous block 310 whether, e.g., the result is equal to or lower than zero. If it is

determined that the object has not expired, the block 306 is performed as above by stepping the pointer to indicate the next slot.

Please amend the paragraph beginning at page 22, line 20, as follows:

If it is determined in the block 316 that the object has expired, a block ~~316~~ 312 is executed in which it is decided whether there has been activity in the object since the previous scanning of the object section 613. It is determined by testing the activity counter to find out whether it is larger than zero. If it is found that there has been activity in the object, the object is only set to being running state, by setting the state field accordingly, in a block 318, the activity counter is reset, not shown, and in the next block 319 the time until expiration is renewed or restarted by setting it equal to the expiration time period. Then the block 306 is executed as described in order to possibly handle a new slot.

Please amend the paragraph beginning at page 23, line 15, as follows:

If it is determined in the block 321 that the object is in a lost state, a block 326 is executed in which an alarm is made, e.g., to an alarm program such as the enterprise management agent 107, also called a management framework or agent which in turn can inform an operator by displaying information on the enterprise management console 108, see Fig. 1. After executing the block 326, the slot storing the last object is marked free, and the block 306 is executed again to possibly handle a new slot.

Please amend the paragraph beginning at page 24, line 5, as follows:

Contexts are sets of data or information. They are defined in the setup of the application or service and make it possible to define code areas holding dynamic data. The introduction of contexts in the monitoring system makes it possible to store session related information when entering execution of certain code areas. The use of contexts makes it easier to correlate faults in objects in a distributed environment. Contexts are stored in the persistent location related to the object, but could be retrieved in XML format and further stored from XML format. This makes it possible to transport contexts between objects and between servers, holding information that could be very important ~~to be~~ to identify the source of a problem or fault that has occurred.

Please amend the paragraph beginning at page 24, line 18, as follows:

The register key created when registering an object can be later used when performing a register, e.g., when registering objects created by a current object to show inheritance. When the register key is used in this way, it possible to see a relation in time between different objects and their execution.